به نام خداوند بخشنده و مهربان

درس ساختمان داده‌ها و الگوریتم‌ها

گزارش پروژه اول

استاد درس: دکتر محمد اکبری

نام دانشجو:

اشکان ودادی گرگری

9713032

زمستان 1398

سوال 2 :range-tree

نحوه پیاده سازی:

ابتدا بعد از گرفتن ورودی ها اقدام سورت کرده و سپس می ایم داده ی محور x ها را به درخت بالانس AVL میدهم. و با y ها هم همین کار را میکنم.

سپس بازه ها را میگیرم و بررسی میکنم که از روت درخت آیا که آن ها را داشته باشد.

5 شرط است برای هر عنصر

```
if(t->key==x2 && t->key > x1)
else if(t->key==x1 && t->key <x2)
else if(t->key>x1 && t->key<x2)
else if(t->key>x1 && t->key>x2)
else if(t->key<x1 && t->key<x2)
```

(راه بهینه تر این بود که y ها هم در class درختمان تعریف میکردم)

سپس هر عمل مثل صف انجام میشود.

در انتها هم چاپ میکنیم.

روند اجرای برنامه:

تست کیس نمونه اول:

```
7
1 9 2 7 3 5 6
1 3 7 2 5 8 6
3
0 0 5 5
8 1 10 5
0 0 10 10
```

در مثال بالا ۷ نقطه وجود دارد: (1,1),(9,3),(2,7),(7,2),(3,5),(5,8),(6,6)

3 بازه به ما داده شده است:

1- $x,y \in [0,5]$
2- $X \in [8,10]$, $y \in [1,5]$
3- $x,y \in [0,10]$

- بعد از گرفتن ورودی ها نقطه ها را در آرایه های تک بعدی x , y ذخیره میکنیم و بر اساس x ها اقدام به سورت کرده:

```
float x[n], y[n];
for(int i = 0 ; i < n ; i ++)
    cin>>x[i];
for(int i = 0 ; i < n ; i ++)
    cin>>y[i];
selectionSort(x,y,n);
```

- برای ساخت AVL:

```
Node *root = NULL;
for(int i = 0 ; i < n ; i ++)
    root = insert(root, x[i]);
```

- برای گرفتن بازه ها: (flagEnter برای کنترل space های موجود است.که دو برابر تعداد بازه ها است)

```
int n2;
cin>>n2;
float x1[n2],x2[n2],y1[n2],y2[n2];
for(int i = 0 ; i < n2 ; i++)
{
    cin>>x1[i];
    cin>>y1[i];
    cin>>x2[i];
    cin>>y2[i];
}
flagEnter = n2*2;
for(int i = 0 ; i < n2 ; i++)
{
    searchOn2DRangeTree(root,x1[i],y1[i],x2[i],y2[i],n,x,y);
}
```

- تابع `searchOn2DRangeTree(root,x1[i],y1[i],x2[i],y2[i],n,x,y)` تمام دستورات ما را برعهده دارد. ورودب هایش ریشه درخت ، لیست x و y های بازه ها و لیست نقاطمان است.

- حال به بررسی درون تابع searchOn2DRangeTree میپردازیم:
- این دستورات برای بررسی بازه ورودی است ، که اگر برعکس دادند درستش کنیم ، در کد من همواره x2>x1 و y2>y1 باید برقرار باشد.

```
if(x1>x2)
{
    int temp = x2;
        x2 = x1;
        x1 = temp;
}
if(y1>y2)
{
    int temp = y2;
        y2 = y1;
        y1 = temp;
}
```

- این قسمت از کد همان صف است که نیاز داریم برای بررسی گره های درخت AVL : list همان صف ما است.
- متغیر K برای نگهداری داده هایی که صدق میکنند است ، آرایه a[n] برای ذخیره سازی آن داده ها است.

```
float a[n];
int k = 0;
list<Node*> list;
list.push_back(rootX);
while(!list.empty())
{
    Node* t = NULL;
    t= list.back();
    list.pop_back();
    if(t->key==x2 && t->key > x1)
    {
        a[k] = t->key;
        if(t->left!=NULL)
            list.push_back(t->left);
        k++;
    }
    else if(t->key==x1 && t->key <x2)
    {
        a[k] = t->key;
        if(t->right!=NULL)
            list.push_back(t->right);
        k++;
    }
    else if(t->key>x1 && t->key<x2)
    {
        a[k] = t->key;
        if(t->right!=NULL)
            list.push_back(t->right);
        if(t->left!=NULL)
            list.push_back(t->left);
        k++;
    }
    else if(t->key>x1 && t->key>x2)
    {
        if(t->left!=NULL)
            list.push_back(t->left);
    }
    else if(t->key<x1 && t->key<x2)
    {
        if(t->right!=NULL)
            list.push_back(t->right);
```

```
    }
}
```

- شرط ها را با مثال بررسی میکنیم:
- مثلا x ها بین 0 تا 5 باشد. اگر root ما 5 باشد شرط اول برقرار است. ریشه در آرایه قرار میگیرد و بچه چپی آن در صف درج مکنیم.
- مثلا x ها بین 10 تا 5 باشد. اگر root ما 5 باشد شرط دوم برقرار است. ریشه در آرایه قرار میگیرد و بچه راستی آن در صف درج مکنیم.
- مثلا x ها بین 0 تا 5 باشد. اگر root ما 3 باشد شرط سوم برقرار است. ریشه در آرایه قرار میگیرد و بچه چپی و راستی آن در صف درج مکنیم.
- مثلا x ها بین 6 تا 8 باشد. اگر root ما 5 باشد شرط پنجم برقرار است. ریشه در آرایه قرار نمیگیرد و بچه راستی آن در صف درج مکنیم.
- مثلا x ها بین 0 تا 5 باشد. اگر root ما 6 باشد شرط چهارم برقرار است. ریشه در آرایه قرار نمیگیرد و بچه چپی آن در صف درج مکنیم.
- شرط حلقه هم تا زمانی که صف خالی نباشد ادامه پیدا میکند.
- پیچیدگی یافتن عنصر در بهترین حالت : logn
- پیچیدگی یافتن عنصر در میانگین حالت : klogn

چالش! :

چالش این سوال برای من در ابتدا فهم سوال بود

بعد از آن هم پیاده سازی AVL برای اولین بار خودش چالشی بود که به کمک اینترنت تا حدودی متوجه آن شدم.

مرحله بعد چالش هم طراحی الگوریتم با کمترین پیچیدگی زمانی بود.

پایان